A METHOD AND APPARATUS FOR REDUCING DISCLOSURE OF PROPRIETARY DATA IN A NETWORKED ENVIRONMENT

FIELD OF THE INVENTION

[0001] The present invention relates to a method and apparatus for accessing resources and, in particular, to a method and apparatus for accessing network resources implementing enhanced security to reduce disclosure of proprietary data in a networked environment.

BACKGROUND OF THE INVENTION

[0002] Conventionally, users on a client system have used a combination of a web browser and other client-based applications to access a content file retrieved from a remote location. For example, the user may access Internet content using INTERNET EXPLORER from Microsoft Corporation of Redmond, Washington and then use WINDOWS EXPLORER also from Microsoft Corporation to access a desktop productivity document type such as a WORD document that has been downloaded to a local location.

[0003] The conventional process requires downloading the file to the client node for viewing and manipulation. However, this

process presents difficulties from a security standpoint. In order to access the content at the client, twice the user is required to save the content locally to non-volatile memory. The first save is required during download and the second is required post editing prior to the uploading process. Additionally, many users frequently move and/or copy the downloaded content from one local directory to another (e.g.: from dir://downloaded_files to dir://my_documents). Each of these save actions creates a local copy of the document on the client. Very few users of the client device will remember to manually delete these local copies of the documents, which accordingly remain on the client device.

[0004] Further, direct manipulation of the client device's storage may be inaccessible to the user, such as the situation where the client device is located in a public kiosk setting. In these cases, the option of deleting the local copy is not available to the user. Since documents left on the client may be accessed by unauthorized individuals with access to the client machine this presents a significant security issue. Additionally, smaller device types, such as personal digital assistants may not have sufficient resources to allow use of client-based applications on the device.

[0005] In an attempt to solve these concerns, conventional methods of access control may require particular authentication credentials from the client prior to granting access and may deny access from inappropriate locations or devices. However, a limitation to conventional methods typically requires that the access control decision result in either a denial or a grant of access to a resource. In the event of a denial, the methods fail to provide any alternative methods of access. In the event of a grant, the methods can provide only full and complete disclosure of the resource. A method of granting access control by assigning degrees of access based on access control levels would be desirable in providing access to proprietary resources in a networked environment.

[0006] Additionally, in protecting proprietary data from improper client node access, it would be desirable for access rights to provide alternative methods of accessing files, depending upon factors such as the client device type, authorization credentials, and capabilities. An alternative to complete denial of access rights, such as limited rights to files executed on a secure network on behalf of the client, would be desirable.

BRIEF SUMMARY OF THE INVENTION

[0007] The present invention relates to a method and apparatus for accessing network resources implementing enhanced security to reduce disclosure of proprietary data in a networked environment.

[0008] In one aspect, the invention relates to a method and apparatus for providing file contents. A client node transmits a request for a file. An access control server receives the request and makes an access control decision. An application server farm presents the contents of the file to the client node using an application program associated with a file type of the requested file.

[0009] In one embodiment, the access control server gathers information about the client node before making the access control decision. In another embodiment, the access control server transmits to the client node an executable containing an identifier for the application program associated with the file type and an identifier for an application server capable of presenting the contents of the file to the client node. In one embodiment, the access control server identifies the application program associated with the file type before presentation of the file contents to the

client node. In another embodiment, the application server identifies the application program associated with the file type before presentation of the file contents to the client node.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] These and other aspects of this invention will be readily apparent from the detailed description below and the appended drawings, which are meant to illustrate and not to limit the invention, and in which:

[0011] FIG. 1A is a block diagram of an environment suitable for practicing the illustrative embodiment of the present invention;

[0012] FIG. 1B and 1C are block diagrams depicting embodiments of computers useful in connection with the present invention;

[0013] FIG. 1D is a block diagram of an embodiment of a computer network in which the network provides a policy-based system of granting access to network resources;

[0014] FIG. 2 is a more detailed block diagram of an embodiment of a policy engine;

[0015] FIG. 3 is a flow diagram depicting one embodiment of the steps taken by a policy engine to make an access control decision based upon information received about a client node;

[0016] FIG. 4 is a block diagram of an embodiment of a computer network in which the network provides policy-based access to file contents for a client node;

[0017] FIG. 4B is a flow diagram depicting one embodiment of the steps taken by an application server farm to provide file contents to a client node;

[0018] FIG. 5 is a block diagram of an embodiment of a computer network in which the network grants access to transformed content of a resource;

[0019] FIG. 6 is a flow diagram depicting one embodiment of the steps taken by a transformation server to transform the content of the requested file and present the transformed contents to a client node;

[0020] FIG. 7 is a block diagram of an embodiment of a computer network in which authorized remote access to a plurality of application sessions is provided; and

[0021] FIG. 7B is a flow diagram depicting one embodiment of the steps taken by a session server to connect a client node with its associated application sessions.

DETAILED DESCRIPTION OF THE INVENTION

[0022] The illustrative embodiment of the present invention is applicable to a distributed networking environment where a remote user requests access to content. Prior to discussing the specifics of the present invention, it may be helpful to discuss some of the network environments in which the illustrative embodiment of the present invention may be employed.

[0023] FIG. 1A is a block diagram of an environment suitable for practicing the illustrative embodiment of the present invention. A client node 102 includes a web browser 110 and application programs 112a, 112b...112n. An application program is any program that processes data to provide output and that uses an operating system for access to system resources. Exemplary application programs include: word processing applications, such as MICROSOFT WORD, manufactured by Microsoft Corporation of Redmond, Washington; spreadsheet programs, such as MICROSOFT EXCEL, manufactured by Microsoft Corporation; electronic mail

programs, such as MICROSOFT OUTLOOK, manufactured by Microsoft Corporation and GROUPWISE, manufactured by Novell Corp. of Provo, Utah; and productivity suites such as STAR OFFICE, manufactured by Sun Microsystems of Mountain View, California.

[0024] A content server 126 includes content files 128 and may be connected to data stores 122 and 130 holding additional content files 124 and 132 respectively. Those skilled in the art will recognize that other network storage devices or document repositories holding content files may also be networked to the content server 126 without departing from the scope of the present invention. A user of the client node 102 may request content from the content server 126 using the web browser 110 to send a request such as the depicted Hypertext Transport Protocol Secure (HTTPS) request 115, or an HTTP (Hypertext Transport Protocol), FTP (File Transport Protocol) request, or, for operations on file shares, SMB (Server Management Block Protocol) request.

[0025] In many embodiments, the content server 126, client node 102, and the proxy server 120 are provided as personal computer or computer servers, of the sort manufactured by the Hewlett-Packard Corporation of Palo Alto, California or the Dell

Corporation of Round Rock, TX. Figures 1B and 1C depict block diagrams of a typical computer 100 useful as the content server 126, the proxy server 120, or the client node 102 in those embodiments. As shown in Figures 1B and 1C, each computer 100 includes a central processing unit 102, and a main memory unit 104. Each computer 100 may also include other optional elements, such as one or more input/output devices 130a-130n (generally referred to using reference numeral 130), and a cache memory 140 in communication with the central processing unit 102.

[0026] The central processing unit 102 is any logic circuitry that responds to and processes instructions fetched from the main memory unit 104. In many embodiments, the central processing unit is provided by a microprocessor unit, such as: the 8088, the 80286, the 80386, the 80486, the Pentium, Pentium Pro, the Pentium II, the Celeron, or the Xeon processor, all of which are manufactured by Intel Corporation of Mountain View, California; the 68000, the 68010, the 68020, the 68030, the 68040, the PowerPC 601, the PowerPC604, the PowerPC604e, the MPC603e, the MPC603ei, the MPC603ev, the MPC603r, the MPC603p, the MPC740, the MPC745, the MPC750, the MPC755, the MPC7400, the

MPC7410, the MPC7441, the MPC7445, the MPC7447, the MPC7450, the MPC7451, the MPC7455, the MPC7457 processor, all of which are manufactured by Motorola Corporation of Schaumburg, Illinois; the Crusoe TM5800, the Crusoe TM5600, the Crusoe TM5500, the Crusoe TM5400, the Efficeon TM8600, the Efficeon TM8300, or the Efficeon TM8620 processor, manufactured by Transmeta Corporation of Santa Clara, California; the RS/6000 processor, the RS64, the RS 64 II, the P2SC, the POWER3, the RS64 III, the POWER3-II, the RS 64 IV, the POWER4, the POWER4+, the POWER5, or the POWER6 processor, all of which are manufactured by International Business Machines of White Plains, New York; or the AMD Opteron, the AMD Athalon 64 FX, the AMD Athalon, or the AMD Duron processor, manufactured by Advanced Micro Devices of Sunnyvale, California.

[0027] Main memory unit 104 may be one or more memory chips capable of storing data and allowing any storage location to be directly accessed by the microprocessor 102, such as Static random access memory (SRAM), Burst SRAM or SynchBurst SRAM (BSRAM), Dynamic random access memory (DRAM), Fast Page Mode DRAM (FPM DRAM), Enhanced DRAM (EDRAM), Extended Data

Output RAM (EDO RAM), Extended Data Output DRAM (EDO DRAM), Burst Extended Data Output DRAM (BEDO DRAM), Enhanced DRAM (EDRAM), synchronous DRAM (SDRAM), JEDEC SRAM, PC100 SDRAM, Double Data Rate SDRAM (DDR SDRAM), Enhanced SDRAM (ESDRAM), SyncLink DRAM (SLDRAM), Direct Rambus DRAM (DRDRAM), or Ferroelectric RAM (FRAM).

[0028] In the embodiment shown in FIG. 1B, the processor 102 communicates with main memory 104 via a system bus 120 (described in more detail below). FIG. 1C depicts an embodiment of a computer system 100 in which the processor communicates directly with main memory 104 via a memory port. For example, in FIG. 1C, the main memory 104 may be DRDRAM.

[0029] FIG. 1B and FIG. 1C depict embodiments in which the main processor 102 communicates directly with cache memory 140 via a secondary bus, sometimes referred to as a "backside" bus. In other embodiments, the main processor 102 communicates with cache memory 140 using the system bus 120. Cache memory 140 typically has a faster response time than main memory 104 and is typically provided by SRAM, BSRAM, or EDRAM.

[0030] In the embodiment shown in FIG. 1B, the processor 102 communicates with various I/O devices 130 via a local system bus 120. Various busses may be used to connect the central processing unit 102 to the I/O devices 130, including a VESA VL bus, an ISA bus, an EISA bus, a MicroChannel Architecture (MCA) bus, a PCI bus, a PCI-X bus, a PCI-Express bus, or a NuBus. For embodiments in which the I/O device is a video display, the processor 102 may use an Advanced Graphics Port (AGP) to communicate with the display. FIG. 1C depicts an embodiment of a computer system 100 in which the main processor 102 communicates directly with I/O device 130b via HyperTransport, Rapid I/O, or InfiniBand. FIG. 1C also depicts an embodiment in which local busses and direct communication are mixed: the processor 102 communicates with I/O device 130a using a local interconnect bus while communicating with I/O device 130b directly.

[0031] A wide variety of I/O devices 130 may be present in the computer system 100. Input devices include keyboards, mice, trackpads, trackballs, microphones, and drawing tablets. Output devices include video displays, speakers, inkjet printers, laser

printers, and dye-sublimation printers. An I/O device may also provide mass storage for the computer system 100 such as a hard disk drive, a floppy disk drive for receiving floppy disks such as 3.5-inch, 5.25-inch disks or ZIP disks, a CD-ROM drive, a CD-R/RW drive, a DVD-ROM drive, tape drives of various formats, and USB storage devices such as the USB Flash Drive line of devices manufactured by Twintech Industry, Inc. of Los Alamitos, California.

[0032] In further embodiments, an I/O device 130 may be a bridge between the system bus 120 and an external communication bus, such as a USB bus, an Apple Desktop Bus, an RS-232 serial connection, a SCSI bus, a FireWire bus, a FireWire 800 bus, an Ethernet bus, an AppleTalk bus, a Gigabit Ethernet bus, an Asynchronous Transfer Mode bus, a HIPPI bus, a Super HIPPI bus, a SerialPlus bus, a SCI/LAMP bus, a FibreChannel bus, or a Serial Attached small computer system interface bus.

[0033] General-purpose desktop computers of the sort depicted in FIG. 1B and FIG. 1C typically operate under the control of operating systems, which control scheduling of tasks and access to system resources. Typical operating systems include: MICROSOFT WINDOWS, manufactured by Microsoft Corp. of

Redmond, Washington; MacOS, manufactured by Apple Computer of Cupertino, California; OS/2, manufactured by International Business Machines of Armonk, New York; and Linux, a freely-available operating system distributed by Caldera Corp. of Salt Lake City, Utah, among others.

[0034] The client node 102 may be any personal computer (e.g., 286, 386, 486, Pentium, Pentium II, Macintosh computer), Windows-based terminal, Network Computer, wireless device, information appliance, RISC Power PC, X-device, workstation, mini computer, main frame computer, personal digital assistant, or other computing device that has a windows-based desktop and sufficient persistent storage for executing a small, display presentation program. The display presentation program uses commands and data sent to it across communication channels to render a graphical display. Windows-oriented platforms supported by the client node 102 can include, without limitation, WINDOWS 3.x, WINDOWS 95, WINDOWS 98, WINDOWS NT 3.51, WINDOWS NT 4.0, WINDOWS 2000, WINDOWS CE, MAC/OS, Java, and UNIX. The client node 102 can include a visual display device (e.g., a computer monitor), a data entry device (e.g., a keyboard), persistent or volatile storage (e.g.,

computer memory) for storing downloaded application programs, a processor, and a mouse. Execution of a small, display presentation program allows the client node 102 to participate in a distributed computer system model (i.e., a server-based computing model).

[0035] For embodiments in which the client node 102 is a mobile device, the device may be a JAVA-enabled cellular telephone, such as the i50sx, i55sr, i58sr, i85s, i88s, i90c, i95cl, or the im11000, all of which are manufactured by Motorola Corp. of Schaumburg, Illinois, the 6035 or the 7135, manufactured by Kyocera of Kyoto, Japan, or the i300 or i330, manufactured by Samsung Electronics Co., Ltd., of Seoul, Korea. In other embodiments in which the client node 102 is mobile, it may be a personal digital assistant (PDA) operating under control of the PalmOS operating system, such as the Tungsten W, the VII, the VIIx, the i705, all of which are manufactured by palmOne, Inc. of Milpitas, California. In further embodiments, the client node 102 may be a personal digital assistant (PDA) operating under control of the PocketPC operating system, such as the iPAQ 4155, iPAQ 5555, iPAQ 1945, iPAQ 2215, and iPAQ 4255, all of which manufactured by Hewlett-Packard Corporation of Palo Alto, California, the

ViewSonic V36, manufactured by ViewSonic of Walnut, California, or the Toshiba PocketPC e405, manufactured by Toshiba America, Inc. of New York, New York. In still other embodiments the client node is a combination PDA/telephone device such as the Treo 180, Treo 270 or Treo 600, all of which are manufactured by palmOne, Inc. of Milpitas, California. In still further embodiments, the client node 102 is a cellular telephone that operates under control of the PocketPC operating system, such as the MPx200, manufactured by Motorola Corp.

[0036] Referring now to FIG. 1D, one embodiment of a computer network 100 constructed in accordance with the invention is depicted, which includes a client node 102, a collection agent 104, a policy engine 106, a policy database 108, an application server farm 114, and an application server 116. Although only one client node 102, collection agent 104, policy engine 106, application server farm 114, and application server 116 are depicted in the embodiment shown in Figure 1D, it should be understood that the system may provide multiple ones of any or each of those components. For example, in one embodiment, the system 100 includes multiple, logically-grouped application server

116, each of which are available to execute applications on behalf of a client node 102. In these embodiments, the logical group of servers may be referred to as a "server farm." In some of these embodiments, the servers may be geographically dispersed.

[0037] In brief overview, when the client node 102 transmits a request 110 to the policy engine 106 for access to a resource, the collection agent 104 communicates with client node 102, retrieving information about the client node 102, and transmits the client node information 112 to the policy engine 106. The policy engine 106 makes an access control decision by applying a policy from the policy database 108 to the received information 112.

[0038] In more detail, the client node 102 transmits a request 110 for a resource to the policy engine 106. In some embodiments, the client node 102 transmits the request 110 over a network connection. The network can be a local area network (LAN), a metropolitan area network (MAN), or a wide area network (WAN) such as the Internet. The client node 102 and the policy engine 106 may connect to a network through a variety of connections including standard telephone lines, LAN or WAN links (e.g., T1, T3, 56 kb, X.25), broadband connections (ISDN, Frame Relay, ATM), and

wireless connections. Connections between the client node 102 and the policy engine 106 may use a variety of data-link layer communication protocols (e.g., TCP/IP, IPX, SPX, NetBIOS, NetBEUI, SMB, Ethernet, ARCNET, Fiber Distributed Data Interface (FDDI), RS232, IEEE 802.11, IEEE 802.11a, IEE 802.11b, IEEE 802.11g and direct asynchronous connections).

[0039] Upon receiving the request, the policy engine 106 initiates information gathering by the collection agent 104. The collection agent 104 gathers information regarding the client node 102 and transmits the information 112 to the policy engine 106.

[0040] In some embodiments, the collection agent 104 gathers and transmits the information 112 over a network connection. In some embodiments, the collection agent 104 comprises bytecode, such as an application written in the bytecode programming language JAVA. In some embodiments, the collection agent 104 comprises at least one script. In those embodiments, the collection agent 104 gathers information by running at least one script on the client node 102. In some embodiments, the collection agent comprises an Active X control on the client node 102. An Active X control is a specialized COM (Component Object Model)

object that implements a set of interfaces that enable it to look and act like a control.

[0041] In some embodiments, the collection agent 104 executes on the client node. In other embodiments, the collection agent 104 resides on the policy engine 106. In still other embodiments, the collection agent 104 resides on a server. In other embodiments, the policy engine 106 resides on the server. In some of these embodiments, the collection agent 104 resides on both the policy engine 106 and the server.

[0042] In one embodiment, the policy engine 106 transmits the collection agent 104 to the client node 102. In one embodiment, the policy engine 106 requires a second execution of the collection agent 104 after the collection agent 104 has transmitted information 112 to the policy engine 106. In this embodiment, the policy engine 106 may have insufficient information 112 to determine whether the client node 102 satisfies a particular condition. In other embodiments, the policy engine 106 requires a plurality of executions of the collection agent 104 in response to received information 112.

[0043] In some embodiments, the policy engine 106 transmits instructions to the collection agent 104 determining the type of information the collection agent 104 gathers. In those embodiments, a system administrator may configure the instructions transmitted to the collection agent 104 from the policy engine 106. This provides greater control over the type of information collected. This also expands the types of access control decisions that the policy engine 106 can make, due to the greater control over the type of information collected. The collection agent 104 gathers information 112 including, without limitation, machine ID of the client node, operating system type, existence of a patch to an operating system, MAC addresses of installed network cards, a digital watermark on the client device, membership in an Active Directory, existence of a virus scanner, existence of a personal firewall, an HTTP header, browser type, device type, network connection information, and authorization credentials.

[0044] In some embodiments, the device type is a personal digital assistant. In other embodiments, the device type is a cellular telephone. In other embodiments, the device type is a laptop

computer. In other embodiments, the device type is a desktop computer. In other embodiments, the device type is an Internet kiosk.

[0045] In some embodiments, the digital watermark includes data embedding. In some embodiments, the watermark comprises a pattern of data inserted into a file to provide source information about the file. In other embodiments, the watermark comprises data hashing files to provide tamper detection. In other embodiments, the watermark provides copyright information about the file.

[0046] In some embodiments, the network connection information pertains to bandwidth capabilities. In other embodiments, the network connection information pertains to Internet Protocol address. In still other embodiments, the network connection information consists of an Internet Protocol address. In one embodiment, the network connection information comprises a network zone identifying the logon agent to which the client node provided authentication credentials.

[0047] In some embodiments, the authorization credentials include a number of types of authentication information, including

without limitation, user names, client names, client addresses, passwords, PINs, voice samples, one-time passcodes, biometric data, digital certificates, tickets, etc. and combinations thereof. After receiving the gathered information 112, the policy engine 106 makes an access control decision based on the received information 112.

[0048] Referring now to FIG. 2, it is a block diagram of one embodiment of a policy engine 200, including a first component 202 comprising a condition database 204 and a logon agent 206, and including a second component 210 comprising a policy database 212. The first component 202 applies a condition from the condition database 204 to information received about client node 102 and determines whether the received information satisfies the condition.

[0049] In some embodiments, the first component 202 and the second component 210 are logically separate but not physically separate. In some embodiments, the first component 202 and the second component 210 are logically and physically separate. In some embodiments, the condition database 204 resides on the first

component 202. In other embodiments, the condition database 204 resides on the second component 210.

[0050] In some embodiments, a condition may require that the client node 102 execute a particular operating system to satisfy the condition. In some embodiments, a condition may require that the client node 102 execute a particular operating system patch to satisfy the condition. In still other embodiments, a condition may require that the client node 102 provide a MAC address for each installed network card to satisfy the condition. In some embodiments, a condition may require that the client node 102 indicate membership in a particular Active Directory to satisfy the condition. In another embodiment, a condition may require that the client node 102 execute a virus scanner to satisfy the condition. In other embodiments, a condition may require that the client node 102 execute a personal firewall to satisfy the condition. In some embodiments, a condition may require that the client node 102 comprise a particular device type to satisfy the condition. In other embodiments, a condition may require that the client node 102 establish a particular type of network connection to satisfy the condition.

[0051] If the received information satisfies a condition, the first component 202 stores an identifier for that condition in a data set 208. In one embodiment, the received information satisfies a condition if the information makes the condition true. For example, a condition may require that a particular operating system be installed. If the client node 102 has that operating system, the condition is true and satisfied. In another embodiment, the received information satisfies a condition if the information makes the condition false. For example, a condition may address whether spyware exists on the client node 102. If the client node 102 does not contain spyware, the condition is false and satisfied.

[0052] In some embodiments, the logon agent 206 resides outside of the policy engine 200. In other embodiments, the logon agent 206 resides on the policy engine 200. In one embodiment, the first component 202 includes a logon agent 206, which initiates the information gathering about client node 102. In some embodiments, the logon agent 206 further comprises a data store. In these embodiments, the data store includes the conditions for which the collection agent may gather information. This data store is distinct from the condition DB 204.

[0053] In some embodiments, the logon agent 206 initiates information gathering by executing the collection agent 104. In other embodiments, the logon agent 206 initiates information gathering by transmitting the collection agent 104 to the client node 102 for execution on the client node 102. In still other embodiments, the logon agent 206 initiates additional information gathering after receiving information 112. In one embodiment, the logon agent 206 also receives the information 112. In this embodiment, the logon agent 206 generates the data set 208 based upon the received information 112. In some embodiments, the logon agent 206 generates the data set 208 by applying a condition from the database 204 to the information received from the collection agent 104.

[0054] In another embodiment, the first component 202 includes a plurality of logon agents 206. In this embodiment, at least one of the plurality of logon agents 206 resides on each network domain from which a client node 102 may transmit a resource request. In this embodiment, the client node 102 transmits the resource request to a particular logon agent 206. In some embodiments, the logon agent 206 transmits to the policy

engine 200 the network domain from which the client node 102 accessed the logon agent 206. In one embodiment, the network domain from which the client node 102 accesses a logon agent 206 is referred to as the network zone of the client node 102.

[0055] The condition database 204 stores the conditions which the first component 202 applies to received information. The policy database 212 stores the policies which the second component 210 applies to the received data set. In some embodiments, the condition database 204 and the policy database 212 store data in an ODBC-compliant database. For example, the condition database 204 and the policy database 212 may be provided as an ORACLE database, manufactured by Oracle Corporation of Redwood Shores, Calif. In other embodiments, the condition database 204 and the policy database 212 can be a Microsoft ACCESS database or a Microsoft SQL server database, manufactured by Microsoft Corporation of Redmond, Wash.

[0056] After the first component 202 applies the received information to each condition in the condition database 204, the first component transmits the data set 208 to second component 210. In one embodiment, the first component 202 transmits only

the data set 208 to the second component 210. Therefore, in this embodiment, the second component 210 does not receive information 112, only identifiers for satisfied conditions. The second component 210 receives the data set 208 and makes an access control decision by applying a policy from the policy database 212 based upon the conditions identified within data set 208.

[0057] In one embodiment, policy database 212 stores the policies applied to the received information 112. In one embodiment, the policies stored in the policy database 212 are specified at least in part by the system administrator. In another embodiment, a user specifies at least some of the policies stored in the policy database 212. The user-specified policy or policies are stored as preferences. The policy database 212 can be stored in volatile or non-volatile memory or, for example, distributed through multiple servers.

[0058] In one embodiment, a policy allows access to a resource only if one or more conditions are satisfied. In another embodiment, a policy allows access to a resource but prohibits transmission of the resource to the client node 102. One of the

policies stored in the policy database 212 might require or forbid automatic connection to disconnected application sessions. Yet another policy might make connection contingent on the client node 102 that requests access being within a secure network. Another policy might require or forbid automatic connection to active application sessions currently connected to a different client node 102. A further policy might only allow connection to application sessions after receiving user approval. Another policy might only allow connection for a predetermined time after disconnection. Still another policy only allows connection to application sessions that include specific applications. One policy might allow viewing only of the transformed contents of a requested file. A policy might allow the viewing of only an HTML version of the requested file. In some embodiments, access to a resource is provided while download of the file to the client node 102 is prevented. This may be accomplished in a number of ways, including: transformation of the file contents into a viewer-only format, transforming the file contents into HTML for viewing by a web browser, use of file type association to open the file using an application hosted by a server in a server farm instead of using an application hosted by the client node 102, or by using a system of

the sort described in US Application serial number 10/931405, the contents of which are incorporated herein by reference.

[0059] In some of the embodiments above, the method and apparatus provide document protection for proprietary information. In these embodiments, the client node cannot access the networked resources unless the policy engine 106 grants the client node 102 permission to access the resources. In one of these embodiments, the policy engine 106 is the single exposed network element, to ensure that the client node 102 must access the policy engine 106 in order to access the networked resources. In another of these embodiments, the URLs used to access the networked resources behind the policy engine 106 are rewritten to prevent direct access by the client node 102. In others of the embodiments above, the method and apparatus enhance the capabilities of the client node to access resource otherwise inaccessible. In some of the embodiments above, the method and apparatus provide both protection of proprietary information and enhanced client node capabilities.

[0060] Referring now to FIG. 3, a flow diagram depicts one embodiment of the steps taken by the policy engine 106 to make an

access control decision based upon information received about a client node 102. Upon receiving gathered information about the client node 102 (Step 350), the policy engine 106 generates a data set based upon the information (Step 352). In some embodiments, the policy engine 106 requests further information about the client node 102 from the collection agent 104. In these embodiments, the policy engine 106 requires more than one execution of the collection agent 104 on the client node 102. In those embodiments, the policy engine 106 generates the data set 208 after receiving the additional requested information. In these embodiments, the policy engine 106 may have insufficient information 112 to determine whether the client node 102 satisfies a particular condition. In others of these embodiments, the conditions may be indeterminate. In some of the embodiments where the conditions are indeterminate, the collection agent could not gather the information required to satisfy the condition.

[0061] The data set 208 contains identifiers for each condition satisfied by the received information 112. Then the policy engine 106 applies a policy to each identified condition within the data set 208. That application yields an enumeration of resources

which the client node 102 may access (Step 354). In one embodiment, the resources comprise proprietary data. In some embodiments, the resources comprise web pages. In other embodiments, the resources comprise word processing documents. In still other embodiments, the resources comprise spreadsheets. In some embodiments, the enumeration includes only a subset of the resources that the client node 102 may access. The policy engine 106 then presents that enumeration to the client node 102. In some embodiments, the policy engine 106 creates a Hypertext Markup Language (HTML) document used to present the enumeration to the client node.

[0062] Referring now to FIG. 4, one embodiment of a computer network 400 constructed in accordance with the invention is depicted, which includes a client node 402, a collection agent 404, an access control server 406, a policy database 408, an application server farm 414, a first application server 416, an application database 418, a second application server 420, and a second application database 422. In some embodiments, there is a network boundary separating the network on which the client node

402 resides from the network on which the access control server 406 and application server farm 414 reside.

[0063] In brief overview, when the client node 402 transmits to the access control server 406 a request 410 for access to a resource, the collection agent 404 communicates with client node 402, retrieving information about the client node 402, and transmitting client node information 412 to access control server 406. In one embodiment, the client node 402 transmits the request 410 after policy engine 106 presents the client node 402 with an enumeration of available resources. The access control server 406 makes an access control decision by applying a policy from the policy database 408 to the received information 412. Finally, the access control server 406 transmits a file type to the application server farm 414 for presentation of the file contents to the client node 402. Additional components of the computer network 400 are omitted and will be described further in FIG. 4B.

[0064] Referring now to FIG. 4B, a flow diagram depicts one embodiment of the steps taken by the access control server 406 and the application server farm 414 to provide file contents to the

client node 402. Part of the application server farm 414 is an application server 416.

[0065] In one embodiment, once the access control server 406 decides to grant the client node 402 access to the requested file, the access control server 406 determines the file type for the requested file (Step 452). In other embodiments, the application server 416 determines the file type for the requested file. In still other embodiments, a server other than the application server 416 or the access control server 406. In some embodiments, the server determining the file type must first retrieve the requested file. In some of those embodiments, the file is located on the same side of the network boundary 424 as the server determining the file type. In others of those embodiments, the file is located on the same side of the network boundary 424 as the client node 402. In these embodiments, the method and apparatus enhance the capabilities of the client node to access resources otherwise inaccessible, but they do not provide document protection for proprietary information.

[0066] In some embodiments, the network boundary 424 physically separates at least two networks. In other embodiments,

the network boundary 424 logically separates at least two networks. In one embodiment, the network boundary 424 is a firewall.

[0067] In one embodiment, the file extension is the file type and the server determining the file type does so by extracting the file extension from the file. In another embodiment, a resource fork is the file type. After determining file type, the server determining the file type transmits the file type to the application server farm 414 for retrieval and presentation to the client node 402 (Step 454).

[0068] The application server 416 receives the file type from the access control server 406. (Step 456). In some embodiments, the application server 416 identifies an application program associated with that file type. In other embodiments, the access control server 406 identifies an application program associated with that file type. In still other embodiments, a server other than the access control server 406 or the application server 416 identifies the application program associated with that file type.

[0069] In one embodiment, the server identifying the application program associated with the file type queries an application database 418 to retrieve an identifier for the application program. In some embodiments, the application database 418 is a

registry file. In embodiments where either the application server 416 or a separate server identify the application type based on the file type, the identifying server then transmits to the access control server 406 the identifier to the application program. In some embodiments, the identifying server transmits the identifier to the access control server 406 over a network connection.

[0070] In some embodiments, neither the access control server 406 nor a separate server need to transmit the file type to the application server 416 to determine the identifier of the associated application program. In one of these embodiments, the application server 416 transmits to the access control server 406 a list of hosted application programs and the file types with which those application programs are associated. In these embodiments, the access control server 406 retrieves from the transmitted list the identifier for the application program associated with the file type.

[0071] When the access control server 406 receives the identifier of the application program, the access control server 406 creates and transmits to the client node 402 an executable file (Step 458). In some embodiments, the executable file contains the identifier of the application program. In some embodiments, the

executable file contains the identifier of an application server in the application server farm 414 that will present the contents of the file to the client node 402. In some embodiments, the same application server 416 that identified the application program to use with the file type will present the contents of the file to the client node 402. In other embodiments, a second application server 420 presents the contents of the file to the client node 402. In one embodiment, the executable file contains both the identifier of the application program and the identifier of an application server in the application server farm 414 what will present the contents of the file to the client node 402. In some embodiments, the executable file enables the client node 402 to connect with an identified server using a presentation-layer protocol such as the Independent Computing Architecture (ICA) protocol, available from Citrix Systems, Inc. of Fort Lauderdale, Florida. In other embodiments, the executable file enables the client node 402 to connect with an identified server using the Remote Desktop Protocol (RDP), manufactured by Microsoft Corporation. In other embodiments, the presentation-layer protocol is wrapped in a higher protocol.

[0072] The client node 402 receives the executable file from the access control server 406. The client node 402 connects to the application server 416 identified in the executable file (Step 460). In one embodiment, the client node 402 connects to the identified application server 416 using the ICA protocol. In another embodiment, the client node 402 connects to the identified application server 416 using RDP.

[0073] The application server 416 selects a format for the presentation of the file contents (Step 462). In other embodiments, the access control server 406 identifies the format used to present the file contents. In those embodiments, the access control server 406 may apply a policy to identify the available formats. In some embodiments, the application server 416 selects the format based upon received information about the client node 402. In other embodiments, the application server 416 selects the format by applying a policy to the received information.

[0074] The application server 416 accepts the client node 402 connection and retrieves the requested file (Step 464). In one embodiment, the application server 416 retrieves the file from a web server. In another embodiment, the application server 416

retrieves the file from a file server. In yet another embodiment, the retrieved file is an email attachment. In this embodiment, the application server 416 retrieves the file from an electronic mail server. In some embodiments, the mail server is a Lotus mail server. In other embodiments, the mail server is an Outlook mail server or an Outlook Web Access mail server.

[0075] The application server 416 then presents the contents of the file to the client node 402 over the connection (Step 468). In one embodiment, the file contents presented comprise an email attachment.

[0076] Referring to FIG. 5, one embodiment of a computer network 500 constructed in accordance with the invention is depicted, which includes a client node 502, a collection agent 504, a policy engine 506, a first component 508, a second component 512, a condition database 510, a policy database 512, a transformation server 516, and a storage element 518. In brief overview, when the client node 502 transmits a request 522 for access to a resource from the policy engine 506, the collection agent 504 communicates with client node 502, retrieving information about the client node 502, and transmitting client node

information 512 to the policy engine 506. The policy engine 506 makes an access control decision as discussed in FIG. 3 above. Once the policy engine 506 decides to grant the client node 502 access to the requested file, the policy engine 506 transmits the request to the transformation server 516 for transformation and presentation to the client node 502.

[0077] In more detail, the policy engine 506 receives a request from the client node 502 for the transformed contents of a file. In one embodiment, the policy engine 506 identifies a transformation server 516 capable of presenting the transformed contents of the file to the client node 502. In some embodiments, the transformation server 516 is capable of presenting the transformed contents of the file because it contains a copy of previously transformed contents. In other embodiments, the transformation server 516 is capable of presenting the transformed contents of the file because it has the capacity to transform the file contents presently.

[0078] In one embodiment, the policy engine 506 identifies a transformation server 516 by querying a storage element 518 to determine whether a transformation server 516 previously

transformed the contents of the file. In that embodiment, the policy engine 506 transmits the identifier of the transformation server 518 identified by the storage element 518 to the client node 502. In other embodiments, no transformation server 516 has previously transformed the contents. In those embodiments, the policy engine identifies instead a transformation server 516 capable of presently transforming the contents of the file and transmits the request of the client node 502 to that transformation server 516.

[0079] In other embodiments, a server other than the policy engine 506 identifies the transformation server 516 capable of presenting the transformed contents of the file to the client. In some of those embodiments, that same server also transmits to the transformation server 516 the request for presentation of the file to the client. In some of these embodiments, the same server identifying the capable transformation server 516 routes transmits the request to the transformation server 516 through a proxy server.

[0080] In one embodiment, the transformation server 516 receives the request from the policy engine 506 for transformation of the contents of a requested file and presentation to the client

node 502. In another embodiment, the transformation server 516 receives the request from the server other than the policy engine 506. The transformation server 516 retrieves the file and transforms the contents from a native format to a second format. The transformation server 516 then accepts a connection from the client node 502 and presents the transformed contents of the file, transforming the contents if not previously transformed. Finally, the transformation server 516 writes to the storage element 518 the identifier of the server transforming the contents of the file and the identifier of the file.

[0081] Referring now to FIG. 6, a flow diagram depicts one embodiment of the steps taken by the transformation server 516 to transform the content of the requested file and present the transformed contents to the client node 502.

[0082] The transformation server 516 receives the request for transformation of the contents of a requested file and presentation to the client node 502 (Step 600). In one embodiment, the transformation server 516 receives this request over a network connection.

[0083] The transformation server 516 transforms the contents of the requested file from a native format into a second format (Step 602). In one embodiment, the transformation server 516 transforms the contents of the file using regular expressions, from a native format into a second format for presentation on the client. In another embodiment, the transformation server 516 transforms the contents of the file into a second format from a native format, which contains a format conversion tool. In another embodiment, the transformation server 516 transforms the contents of the file from a native format into HTML. In another embodiment, the transformation server 516 transforms the contents of the file from a native format into a second format where the second format enables presentation on a personal digital assistant. In another embodiment, the transformation server 516 transforms the contents of the file from a native format into a second format, where the second format enables presentation on a cellular phone. In another embodiment, the transformation server 516 transforms the contents of the file from a native format into a second format, where the second format enables presentation on a laptop computer. In another embodiment, the transformation server 516 transforms the contents of the file from a native format

into a second format, where the second format enables presentation at an Internet kiosk.

[0084] The transformation server 516 writes identifying information about the transformation to the storage element 518 (Step 604). In one embodiment, the identifying information includes an identifier for the transformation server 516 and an identifier for the transformed file. In some embodiments, the identifying information includes a temporary file containing the transformed contents of the file. In those embodiments, the storage element 518 functions as a global cache of transformed file contents.

[0085] After the policy engine 506 identifies the transformation server 516 capable of presenting the transformed contents of the file for the client node 502, the policy server 506 transmits the identifier of the transformation server 516 to the client node 502. The client node 502 receives the identifier and connects to the transformation server 516. The transformation server 516 accepts the connection and presents the transformed contents of the requested file to the client node 502 over the connection (Step 606). In one embodiment, the transformation

server 516 retains the transformed contents of the requested file after the presentation to the client node 502.

[0086] Referring to FIG. 7, one embodiment of a computer network 700 constructed in accordance with the invention is depicted, which includes a first client node 702, a collection agent 704, an policy engine 706, a policy database 708, a condition database 710, a second client node 716, a session server 720, a stored application database 722, an application server farm 724, a first application server 726, a first database 728, a second application server 730, and a second database 732. In brief overview, when the first client node 702 transmits to the access control server 706 a request 712 for access to a resource, the collection agent 704 communicates with client node 702, retrieving information about client node 702, and transmitting client node information 714 to the policy engine 706. The policy engine 706 makes an access control decision, as discussed above in FIG. 3. Finally, the session server 720 establishes a connection between the client node 702 and a plurality of application sessions associated with the client node 702. Additional components of the computer network 700 are omitted and will be described further in FIG. 7B.

[0087] Referring now to FIG. 7B, a flow diagram depicts one embodiment of the steps taken by the session server 720 to connect the client node 702 with its associated application sessions. The session server 720 receives information about the client node 702 from the policy engine 706 containing access control decision the policy engine 706 made. In one embodiment, the information also includes the client node information 714.

[0088] In some embodiments, the policy engine 706 identifies a plurality of application sessions already associated with the client node 702. In other embodiments, the session server 720 identifies stored application sessions associated with the client node 702. In some of these embodiments, the session server 720 automatically identifies the stored application sessions upon receiving the information from the policy engine 706. In one embodiment, the stored application database 722 resides on the session server 720. In another embodiment, the stored application database 722 resides on the policy engine 706.

[0089] The stored application database 722 contains data associated with a plurality of servers in the application server farm 724 executing application sessions. In some embodiments,

identifying the application sessions associated with the client node 702 requires consulting stored data associated with one or more servers executing application sessions. In some of these embodiments, the session store 720 consults the stored data associated with one or more servers executing application sessions. In others of these embodiments, the policy engine 706 consults the stored data associated with one or more servers executing application sessions. In some embodiments, a first application session runs on a first application server 726 and a second application session runs on a second application server 730. In other embodiments, all application sessions run on a single application server within the application server farm 724.

[0090] The session server 720 includes information related to application sessions initiated by users. The session server can be stored in volatile or non-volatile memory or, for example, distributed through multiple servers. Table 7-1 shows the data included in a portion of an illustrative session server 720.

[0091]

Table 7-1

Application Session	App Session 1	App Session 2	App Session 3
User ID	User 1	User 2	User 1
Client ID	First Client		First Client
Client Address	172.16.0.50		172.16.0.50
Status	Active	Disconnected	Active
Applications	Word Processor	Data Base	Spreadsheet
Process Number	1	3	2
Server	Server A	Server A	Server B
Server Address	172.16.2.55	172.16.2.55	172.16.2.56

T[0092] he illustrative session server 720 in Table 7-1 includes data associating each application session with the user that initiated the application session, an identification of the client computer 702 or 716, if any, from which the user is currently connected to the server 726, and the IP address of that client computer 702a or 716. The illustrative session server 720 also includes the status of each application session. An application session status can be, for example, “active” (meaning a user is connected to the application session), or “disconnected” (meaning a

user is not connected to the application session). In an alternative embodiment, an application session status can also be set to "executing-disconnected" (meaning the user has disconnected from the application session, but the applications in the application session are still executing), or "stalled-disconnected" (meaning the user is disconnected and the applications in the application session are not executing, but their operational state immediately prior to the disconnection has been stored). The session server 720 further stores information indicating the applications 116 that are executing within each application session and data indicating each application's process on the server. In embodiments in which the server 726 is part of a server farm 724, the session server 720 is at least a part of the dynamic store, and also includes the data in the last two rows of Table 1 that indicate on which server in the server farm each application is/was executing, and the IP address of that server. In alternative embodiments, the session server 720 includes a status indicator for each application in each application session.

[0093] For example, in the example of Table 7-1, three application sessions exist, App Session 1, App Session 2, and App Session 3. App Session 1 is associated with User 1, who is currently

using terminal 1. Terminal one's IP address is 152.16.2.50. The status of App Session 1 is active, and in App Session 1, a word processing program, is being executed. The word processing program is executing on Server A as process number 1. Server A's IP address is 152.16.2.55. App Session 2 in Table 1 is an example of a disconnected application session 118. App Session 2 is associated with User 2, but App Session 2 is not connected to a client computer 702a or 716. App Session 2 includes a database program that is executing on Server A, at IP address 152.16.2.55 as process number 3. App Session 3 is an example of how a user can interact with application sessions operating on different servers 726. App Session 3 is associated with User 1, as is App Session 1. App Session 3 includes a spreadsheet program that is executing on Server B at IP address 152.16.2.56 as process number 2, whereas the application session included in App Session 1 is executing on Server A.

[0094] In one embodiment, the session server 720 is configured to receive a disconnect request to disconnect the application sessions associated with the client node 702 and does so disconnect the application sessions in response to the request.

The session server 720 continues to execute an application session after disconnecting the client node 702 from the application session. In this embodiment, the session server 720 accesses the stored application database 722 and updates a data record associated with each disconnected application session so that the record indicates that the application session associated with the client node 702 is disconnected.

[0095] Unintentional termination of application sessions resulting from imperfect network connections and users' failure to terminate their application sessions themselves can lead to user difficulties. One embodiment of the invention limits these difficulties by differentiating disconnection (which is treated as if the user is not done working with an application session) from termination (which is assumed to be an intentional end to the application session) and by correlating application sessions with users as opposed to client nodes. When a user is finished using an application operating in an application session, the user can terminate an application session. Termination generally involves the affirmative input of the user indicating that the server should no longer maintain the application session. Such affirmative user input

can include selecting an "Exit" option from a menu, clicking on an icon, etc. In response to the session server 720 receiving a termination request, the execution of the application session and any application within that application session is halted. In one embodiment, data related to the application session is also removed from the stored application database 722.

[0096] Disconnection, either intentional or unintentional, on the other hand, does not result in termination of application sessions. Since the application or applications operating in an application session are executing on the server 720, a connection to the first client node 702 is not usually necessary to continue execution of the applications, and in one embodiment the applications can continue to execute while waiting for the user to connect. In an alternative embodiment, upon disconnection of a user, the session server 720 stalls the execution of the applications operating in the application session. That is, the session server 720 halts further execution of the applications, and the session server 720 stores the operational state of the application and any data the application is processing. In a further embodiment, the session server 720 can selectively stall execution of specific applications

after a user disconnects. For example, in one embodiment, the session server 720 continues execution of an application for a fixed time period, and if a user fails to connect within that time period, the session server 720 stalls the application. In another embodiment, the session server 720 stalls specified application sessions that cannot continue executing without user input. In each of the above-described embodiments, if the user of the first client node 702 disconnects from the server 726 and then connects to the server 726 while operating the first client node 702, the second client node 716, or a third client computer, the session server 720 can connect the client computer operated by the user to one or more previously initiated, non-terminated application session(s) associated with the user, and reinitiate execution of any stalled applications.

[0097] In one embodiment, the session server 720 detects a disconnection. A user can intentionally and manually instruct the server to disconnect an application session from the client node 702 or 716 that the user is communicating from. For example, in one embodiment, application sessions provide a menu option for disconnection (as distinguished from termination above) that a user

can select. The session server 720 can also detect an unintentional disconnection. For example, in one embodiment, session server 720 identifies when a predetermined number of data packets transmitted to a client node 702 or 716 have not been acknowledged by the client node 702 or 716. In another embodiment, the client node 702 or 716 periodically transmits a signal to the server 726 to confirm that a connection is still intact. If the session server 720 detects that a predetermined number of expected confirmation signals from a client node 702 or 716 have not arrived, session server 720 determines that the client node 702 or 716 has disconnected. If the session server 720 detects that a user has disconnected from an application session, either intentionally, or unintentionally, the entry in the session server 720 related to the disconnected application session is modified to reflect the disconnection.

[0098] After receiving authentication information, the session server 720 consults the stored applications database 722 to identify any active application sessions that are associated with the user, but that are connected to a different client node, such as the first client node 702, for example. In one embodiment, if the

session server 720 identifies any such active application sessions, the session server 720 automatically disconnects the application session(s) from the first client node 702 and connects the application session(s) to the current client computer 716. In some embodiments, the received authentication information will restrict the application sessions to which the client node 702 may reconnect. In one embodiment, the user can trigger the automatic consultation of the session server and subsequent connection with the selection of a single user interface element.

[0099] After identifying the application sessions associated with the client node 702, the session server 720 connects the client node 702 to associated application sessions. The session server 720 determines whether each application session in the plurality is active or disconnected. In one embodiment, at least one application session in the plurality is active. In one embodiment, at least one application session in the plurality is disconnected. In one embodiment, the session server 720 receives the application output automatically. In another embodiment, receipt of the application output is triggered by client node 702 selection of a single user interface element. The session server 720 identifies disconnected

application sessions to which to reconnect the client node 702 based upon the access control decision contained in the received information 714. In one embodiment, upon identifying any disconnected application sessions, the session server 720 prompts the user to indicate whether connection is desired. If connection is not desired, the session server 720 prompts user to indicate whether the disconnected applications sessions should remain disconnected, or whether the application sessions should be terminated.

[0100] In one embodiment, connection includes modifying the entry in the stored applications database 722 to indicate that the user is connected to the application session and to indicate from which client node 702 the user is connected to the server. Upon connection, the server 726 resumes transmitting application output data to the client node 702 or 716. In one embodiment, the plurality of application sessions associated with the client node was connected to the first client node 702 prior to connection and, after connection the plurality of application sessions is reconnected to the first client node 702. In another embodiment, the plurality of application sessions associated with the client node was connected

to the first client node 702 prior to connection and, after connection the plurality of application sessions is reconnected to the second client node 716.

[0101] The following illustrative examples show how the methods and apparatus discussed above can be used to provide policy-based access to file contents for a client node. These examples are meant to illustrate and not to limit the invention.

[0102] Evidence Collection

[0103] In one embodiment, a client node 102 requests access to a word processing document located on a server residing on the same network as the policy engine 106 resides. The policy engine 106 receives the request and determines that it possesses no information about client node 102. The policy engine 106 transmits a collection agent 104 to the client node 102. In some embodiments, the collection agent 104 has pre-defined information to collect from the client node. In other embodiments, the collection agent 104 first analyzes the client node to determine what type of information to collect. In still other embodiments, the collection agent 104 retrieves from the policy engine 106 the

instructions as to what information to collect about the client node 102.

[0104] Once executing on the client node 102, the collection agent 104 gathers the required information and transmits the information 112 to the policy engine 106. The policy engine 106 receives the information 112 and begins the process of determining what conditions the information 112 satisfies. In some embodiments, the policy engine 106 determines that the received information 112 does not suffice to determine whether the information 112 satisfies one or more conditions. In those embodiments, the policy engine 106 transmits further instructions to the collection agent 104 for gathering more information about the client node 102.

[0105] Policy-Based Access Control

[0106] As the first component 202 of the policy engine 106 determines that one or more conditions are satisfied, it stores an identifier for each satisfied condition in a data set. Upon completion, the first component 202 transmits the data set and the requested application to the second component 210. In an example of this embodiment, the requested application may be a word

processing document and the conditions satisfied may indicate that the client device is a personal digital assistant. In another example of this embodiment, the requested application may be a spreadsheet and the conditions satisfied may indicate that the client device is a trusted laptop connecting from an insecure network such as a public internet kiosk. In a third example of this embodiment, the requested application may be a file attached to an electronic mail message and the conditions satisfied may indicate that the client device is on a personal desktop connecting from a secure network but lacking the appropriate application software to view the file.

[0107] The second component 210 receives the data set from the first component 202 and applies one or more policies to the received data. In one example of this embodiment, the second component 210 may apply a policy requiring that when a client device type is a personal digital assistant if the condition that the client node have on it application software is not satisfied, the client node receive the transformed contents of the file. The client node would then receive an executable file enabling connection to a transformation server, which will present the contents of the file in

a format accessible to the client device type. Applying this policy enables the client node to view the contents of the file in spite of inappropriate form factor for viewing

[0108] In another example of this embodiment, the second component 210 may apply a policy prohibiting download to the client node 102 when a client device type is a trusted laptop, containing the appropriate application software, but from an insecure network such as an Internet kiosk. In this embodiment, the policy might require that the policy engine 106 transmit an executable file to the client node 102 enabling connection to an application server 416 for presentation of the file contents. Applying a policy of this type, and retrieving the file only to the application server 416, enables the client node 102 to view the contents of the file without jeopardizing the proprietary contents of the file from inappropriate dissemination.

[0109] In yet another example of this embodiment, the second component 210 may apply a policy requiring that a personal desktop making a secure connection, but lacking appropriate application software, connect to an application server 416 via an ICA session, and that the application server 416 execute the

appropriate application and present the file to the client node 102. Applying the policy enables the client node 102 to view the contents of the file regardless of the lack of application software on the client node 102.

[0110] The present invention may be provided as one or more computer-readable programs embodied on or in one or more articles of manufacture. The article of manufacture may be a floppy disk, a hard disk, a compact disc, a digital versatile disc, a flash memory card, a PROM, a RAM, a ROM, or a magnetic tape. In general, the computer-readable programs may be implemented in any programming language. Some examples of languages that can be used include C, C++, C#, or JAVA. The software programs may be stored on or in one or more articles of manufacture as object code.

[0111] While the invention has been shown and described with reference to specific preferred embodiments, it should be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the invention as defined by the following claims.